ARCOS – Architecture and Messaging Guide

1. Purpose of this Guide

This document explains the messaging flow and architecture of ARCOS (AI Rule-Constrained Orchestration System). It consolidates multiple source diagrams and specifications into a single, coherent reference. It is intended to help readers understand how ARCOS agents interact, what artifacts they exchange, and how schemas define those interactions.

2. Architecture Overview

ARCOS is built on an orchestrated loop involving the following roles:

- User – initiates and refines projects.
- Orchestrator (Maestro) – central coordinator of all agent interactions.
- Speculus (ARCOS and Domain) – extracts specifications and rules from user input or domain context.
- Producer – generates domain-specific outputs (e.g., code, configuration, data).
- Validator – checks outputs against domain schemas and rule sets.
- Post-Processor – finalizes, compiles, or packages outputs.
- Filter-Inputs – remove non relevant files from output.zip before sending to the producer using the reports the orchestrator received.

The architecture enforces strict schema validation at every exchange. Each artifact exchanged is represented by XML and validated against an XSD schema. This guarantees structural consistency and early error detection.

3. Messaging Lifecycle

The orchestrator interacts with agents through well-defined request and response messages. Each interaction is defined by its own schema. The key lifecycle is as follows:

1. Project Initialization – The orchestrator sends a NewProject message to the ARCOS Speculus.
2. Domain Rules – The Domain Speculus produces Domain_Rules.xml (validated against Domain_Rules.xsd).
3. Producer Request – The orchestrator sends ProjectSpec, DomainSchema, and rules to the Producer, and in retries, sends the Filtered previous output.zip and the error reports.

4. Producer Response – The Producer returns either an Output (ZipBase64 + checksum) or Clarifications.
5. Validation – The orchestrator sends artifacts to the Validator, which returns a ValidatorReport.xml or Clarification.
6. Retry Loop – If issues exist, the orchestrator may trigger a retry with updated inputs and validation report or post processor report.
7. Post-Processing – Once validated, outputs are refined or compiled, producing a PostProcessorReport.xml.
8. On Post Processor success, the output.zip from the complete project is rebuilt using the Filter-Inputs is sent back to the loop (validation, post-processing) to verify that the full project is now valid all the way.

4. Key Artifacts

The following schemas define the main messages and artifacts in ARCOS:

- ARCOS_Project.xsd – Defines a project, its agents, licensing, and constraints.
- Maestro_ARCOS_Speculus_Messaging.xsd and ARCOS_Speculus_Response.xsd – Defines communication between orchestrator and ARCOS Speculus.
- Maestro_Domain_Speculus_Messaging.xsd and Domain_Speculus_Response.xsd – Defines communication with Domain Speculus.
- Maestro_Producer_Messaging.xsd and Producer_Response.xsd – Define request and response of Producer agent.
- Maestro_Validator_Messaging.xsd, Validator_Response.xsd, and Validator_Report.xsd – Define validation requests, responses, and report.
- Maestro_PostProcessor_Messaging.xsd and Post_Processor_Response.xsd – Define final processing messages and results.
- Post_Processor_Report.xsd – Defines compiler output or final diagnostics.
- Domain_Rules.xsd and Predefined_Domain_Rules.xsd – Define rule structures.
- BLEU_parts_v5.xsd – Example domain schema for inventory management.

5. End-to-End Flow

1. User provides high-level input.
2. ARCOS Speculus captures project-level description and the domain specific agents to use.
3. Domain Speculus extracts or edits domain-specific rules.
4. Producer generates initial outputs.
5. Validator ensures compliance with domain schemas and rules.
6. Post-Processor compiles or packages validated outputs.
7. On errors, Filter the output.zip with only the parts mentioned in the error messages.
8. Reports are returned to the user and linked to original project ID.

6. Example Walkthrough

In the BLEU inventory domain, the orchestrator starts with an ARCOS_Project.xml referencing BLEU_parts_v5.xsd. The Domain Speculus generates Domain_Rules.xml and uses some Predefined_Domain_Rules.xml. The Producer then creates Rust CRUD code packaged as Output.zip. The Validator checks the generated code against the schema and rules, issuing a ValidatorReport.xml. Finally, the Post-Processor compiles the code and issues a PostProcessor_Report.xml with build diagnostics. If any of the validation or post processing reports an error, the Orchestrator does a retry loop sending the rules, the filtered previous output and error report to the producer to try it again. The filtering removes any part of the output.zip that is not mentioned in the reports as being in error.

7. Conclusion

ARCOS messaging and architecture provide a repeatable, auditable process for AI-driven system generation. By constraining communication through schemas, ARCOS ensures that agents remain interoperable, transparent, and trustworthy. This guide consolidates the messaging lifecycle, artifacts, and architectural roles into one reference for implementers and researchers.